Computer Science Department

University of Verona

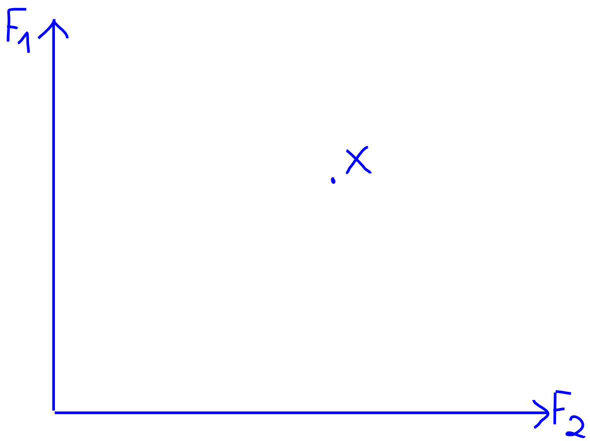
A.A. 2016-17

Pattern Recognition

Feature extraction: PCA, eigenfaces

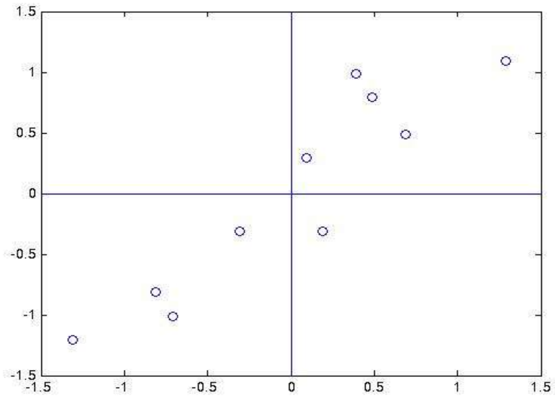
Si considerino due features F1, F2, ed un punto X sul piano creato da queste, che viene descritto dalle sue coordinate (valori delle sue features) t.c. X=(x1,x2)'. Posso riscrivere X del grafico come:

$$X = m + \sum_{i=1}^{2} \bar{e}_i^T x_i = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \bar{e}_1^T x_1 + \bar{e}_2^T x_2 = \begin{vmatrix} 0 \\ 0 \end{vmatrix} + \begin{vmatrix} 1 \\ 0 \end{vmatrix} x_1 + \begin{vmatrix} 0 \\ 1 \end{vmatrix} x_2$$



Principal Component Analysis

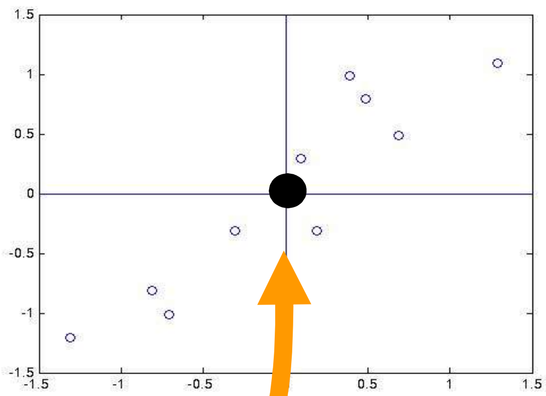
- Il dato = un vettore N-dimensionale di valori di pixel
- Supponiamo N=2 → dati = punti 2D



Come posso descrivere in modo compatto questi M punti?

- SOL 1: Con un punto (vettore) solo, che minimizzi la distanza media quadratica con tutti i pti

$$\underset{\mathbf{x}^{(0)}}{\operatorname{argmin}}\ J_0(\mathbf{x}^{(0)}) = \sum_{k=1}^{M} \left\| \mathbf{x}^{(0)} - \mathbf{x}^{(k)} \right\|^2$$



- Soluzione: il vettore media

$$\mathbf{m} = \frac{1}{M} \sum_{k=1}^{M} \mathbf{x}^{(k)}$$

Problema

m poco espressivo!!!

Descrittore povero, fa sparire l'informazione sulla distribuzione

PCA

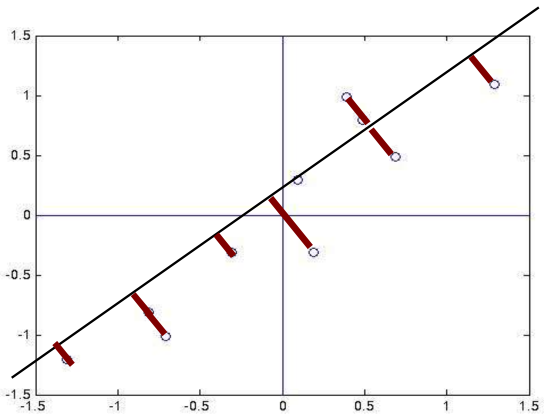
Vuole trovare i valori di rette e proiezioni
che soddisfino determinate caratteristiche.

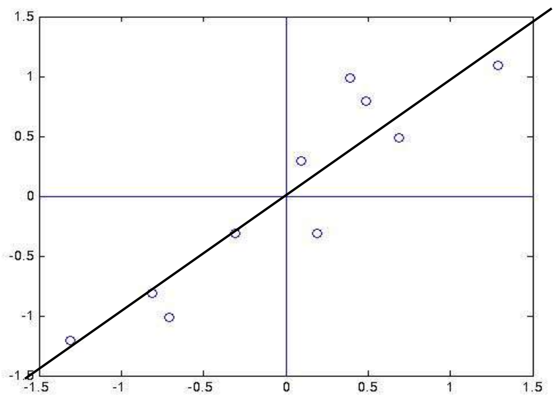
- SOL2: Con una retta di proiezione, che minimizzi la distanza media quadratica con tutti i pti

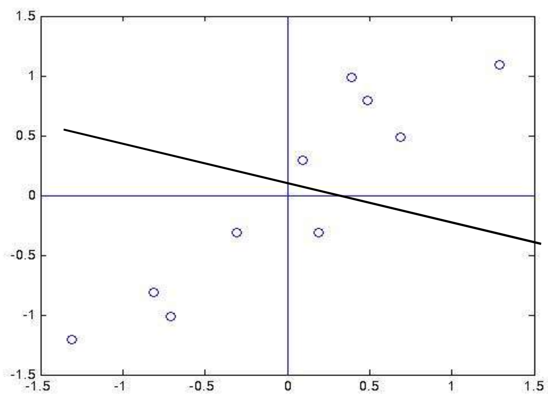
Ogni punto del mio training set dev'essere rappresentato da un'unica feature nuova. Ad esempio scegliamo come feature la distanza da una retta che approssima i punti.

- Supp. una retta di proiezione nello spazio

Alcune vanno bene intuitivamente ... altre no.







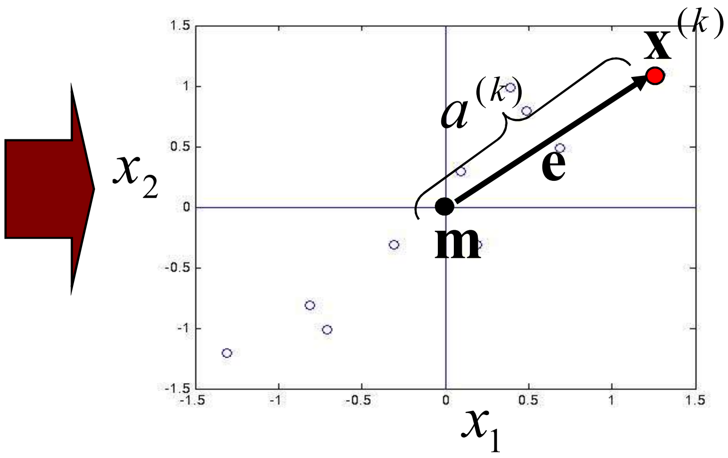
NB: data la matrice di covarianza, maggiore è l'autovalore, maggiore è la diffusione sull'autovettore relativo.

PCA

- Per convenienza, imponiamo passaggio dalla media, e specifichiamo quindi i punti della retta come

$$\mathbf{x}^{(k)} = \mathbf{m} + a^{(k)}\mathbf{e}$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} + a\begin{bmatrix} e_1 \\ e_2 \end{bmatrix}$$



e= versore

- Troviamo quindi i coefficienti $\{a^{(k)}\}_{k=1\ldots M}$ ed i vettori e che minimizzano la distanza quadratica

$$\boxed{\underset{\{a^{(k)}\}_{k=1\ldots M}}{\operatorname{argmin}} J_1\left(\{a^{(k)}\}_{k=1\ldots M}, \mathbf{e}\right) = \sum_{k=1}^{M} \left\| \left(\mathbf{m} + a^{(k)}\mathbf{e}\right) - \mathbf{x}^{(k)} \right\|^2}$$

Sommatoria tra la differenza delle proiezioni dei punti sulla retta di fit e le coordinate originali dei punti stessi

$$\ldots \rightarrow a^{(k)} = \mathbf{e}^T\left(\mathbf{x}^{(k)} - \mathbf{m}\right)$$

Dal funzionale sopra vado a vedere cosa vale il singolo a^k in funzione di e, così da eliminare gli a^k dal funzionale.

PCA

- Sostituendo $a^{(k)} = \mathbf{e}^T\left(\mathbf{x}^{(k)} - \mathbf{m}\right)$ in $J_1\left(\left\{a^{(k)}\right\}_{k=1\ldots M}, \mathbf{e}\right)$ otteniamo $J_1(\mathbf{e})$, ossia

a meno di un fattore di normalizzazione

Scatter matrix ≈ mat. di covarianza

$$J_1(\mathbf{e}) = -\mathbf{e}^T\mathbf{S}\mathbf{e} + \sum_{k=1}^{M}\left\|\mathbf{x}^{(k)} - \mathbf{m}\right\|^2$$

$$\mathbf{S} = \sum_{k=1}^{M}\left(\mathbf{x}^{(k)} - \mathbf{m}\right)\left(\mathbf{x}^{(k)} - \mathbf{m}\right)^T$$

- Minimizzare $J_1(\mathbf{e})$ significa massimizzare $\mathbf{e}^T\mathbf{S}\mathbf{e}$ tenuto conto che $\|\mathbf{e}\| = 1$, via moltiplicatori di Lagrange; ossia $u = \mathbf{e}^T\mathbf{S}\mathbf{e} - \lambda\left(\mathbf{e}^T\mathbf{e} - 1\right)$

- Minimizzo;

$$\frac{\partial u}{\partial \mathbf{e}} = 2\mathbf{S}\mathbf{e} - 2\lambda\mathbf{e} = 0 \Rightarrow \mathbf{S}\mathbf{e} = \lambda\mathbf{e} \Rightarrow$$

A conti fatti:

1. **e** deve essere autovettore;
2. Poiché

$$\mathbf{e}^T\mathbf{S}\mathbf{e} = \lambda\mathbf{e}^T\mathbf{e} = \lambda$$

λ deve essere massimo

devo prendere l'autvec e ad autval max

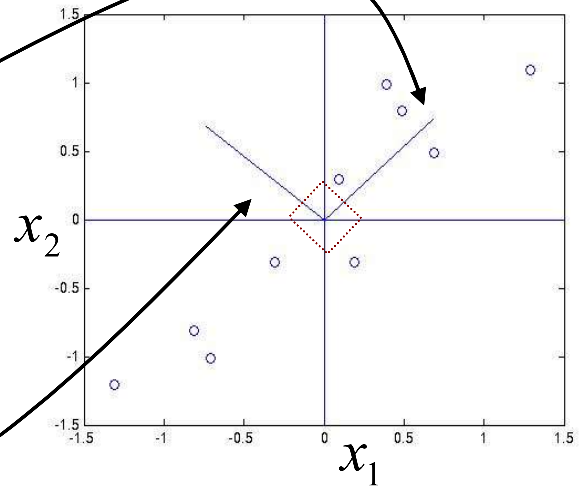
Esempio

- La matrice di covarianza è $S = \begin{bmatrix} 0.617 & 0.615 \\ 0.615 & 0.717 \end{bmatrix}$

- Ottengo 2 autovettori (e autovalori): quale prendo?

$$\mathbf{e}_{(1)} = \begin{bmatrix} -0.735 \\ 0.678 \end{bmatrix} \quad \mathbf{e}_{(2)} = \begin{bmatrix} -0.678 \\ 0.735 \end{bmatrix}$$

$$\lambda_{(1)} = 0.049 \quad \lambda_{(2)} = 1.284$$



Esempio - osservazioni

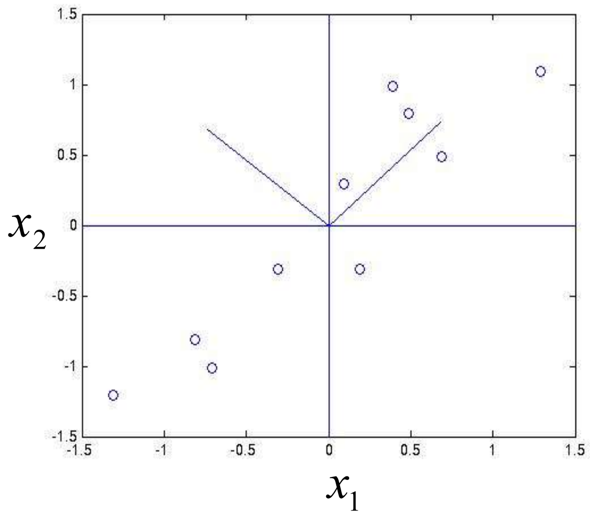
- La matrice di covarianza $S = \begin{bmatrix} 0.617 & 0.615 \\ 0.615 & 0.717 \end{bmatrix}$ è simmetrica, reale

→ gli autovettori {**e**} sono ortogonali, formanti una base per lo spazio N dimensionale

$$\mathbf{e}_1 = \begin{bmatrix} -0.735 \\ 0.678 \end{bmatrix} \quad \mathbf{e}_2 = \begin{bmatrix} -0.678 \\ 0.735 \end{bmatrix}$$

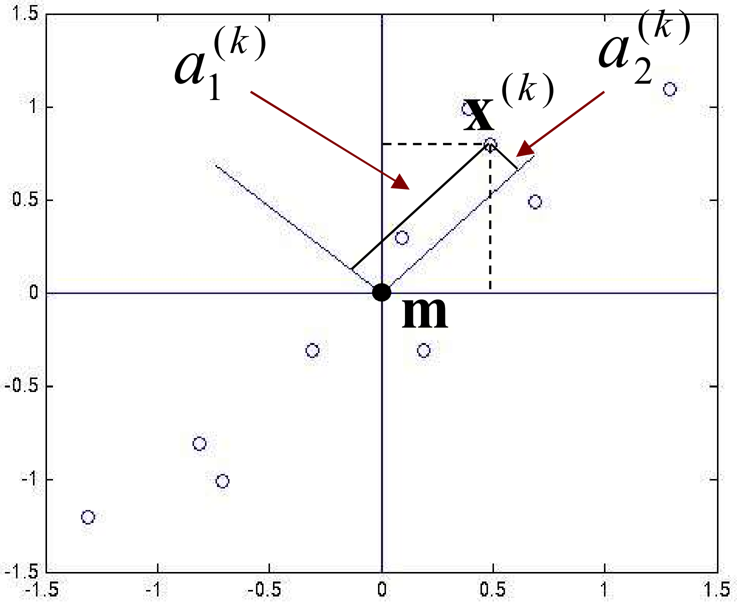
$$\lambda_1 = 0.049 \qquad \lambda_2 = 1.284$$

→ gli autovalori più grandi si hanno nelle direzioni di massima dispersione dei dati



Esempio - osservazioni

- I coefficienti $a_i^{(k)}$, per i =1,...,D, sono le componenti del punto $\mathbf{x}^{(k)}$ per la base trovata



a^(k) è la proiezione rispetto all'origine proiettata sulla base trovata, i.e. PCA può essere vista come un cambio di base dove la nuova base è più efficace in termine di descrizione di punti perché si avvicina più alla distribuzione dei punti.

- Quindi un punto può essere scritto come

$$\boxed{\mathbf{x}^{(k)} = \mathbf{m} + \sum_{i=1}^{D} a_i^{(k)} \mathbf{e}_i}$$

NB: gli autovettori hanno ovviamente la stessa dimensionalità dello spazio a cui afferiscono. I.e. sono punti in quello spazio (nr. pixel)

Da PCA ad eigenfaces

- Dato un gruppo di punti, ho trovato la base che descrive lo scattering dei punti
- Ogni punto puo' essere mappato tramite i componenti della base (numero componenti = numero dimensioni N)
- Usare meno componenti (le componenti principali) permette di proiettare i punti in un sottospazio altamente informativo → riduzione della dimensionalità dei punti
- Ora passiamo alle facce (...punti!)→dataset di M facce, N dimensioni

Posso immaginare di concatenare gli M vettori ottenendo una matrice di facce MxN, dove ogni colonna rappresenta una faccia

$$\mathbf{x}^{(1)} = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ \vdots \\ x_N^{(1)} \end{bmatrix}$$

$$\mathbf{x}^{(2)} = \begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ \vdots \\ x_N^{(2)} \end{bmatrix} \quad \cdots$$

$$\mathbf{x}^{(M)} = \begin{bmatrix} x_1^{(M)} \\ x_2^{(M)} \\ \vdots \\ x_N^{(M)} \end{bmatrix}$$

N.B.:
di solito,
M<<N

Da PCA ad eigenfaces (2)

- Preprocessing -
  - Le immagini devono contenere esclusivamente facce (no outliers)
  - Le immagini devono essere ragionevolmente prive di rumore i.e. devo capire che si tratta di una faccia, il dato dev'essere sufficiente
  - Le facce devono avere la stessa dimensione→ riscalamento
  - ABBASTANZA Stesse condizioni di illuminazione, o compensazione PCA è in grado di compensare abbastanza diverse illuminazioni
  - Non ci deve essere sfondo→ritaglia le facce, catturale su sfondo neutro (alternativa: background subtraction) Posso altrimenti utilizzare algoritmi di face detection
  - In maniera automatica, utilizzo di un metodo di face detection
  - La posa dei volti deve essere il piu' possibile simile (no fronte e profilo assieme)

 $\mathbf{x}^{(1)} = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ \vdots \\ x_N^{(1)} \end{bmatrix}$  $\mathbf{x}^{(2)} = \begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ \vdots \\ x_N^{(2)} \end{bmatrix}$ $\cdots$  $\mathbf{x}^{(M)} = \begin{bmatrix} x_1^{(M)} \\ x_2^{(M)} \\ \vdots \\ x_N^{(M)} \end{bmatrix}$

Eigenfaces - proiezioni

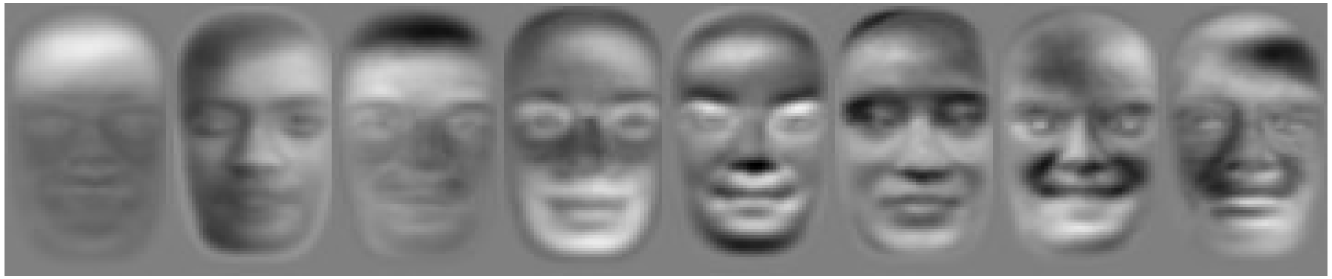
- Dopo aver ricavato gli autovettori {**e**} (eigenfaces) dalla matrice di covarianza S, calcolo le componenti {a} (o proiezione) per la faccia **x**

$$\langle \underbrace{\mathbf{e}'_1(\mathbf{x}-\mathbf{m})}_{a_1}, \underbrace{\mathbf{e}'_2(\mathbf{x}-\mathbf{m})}_{a_2}, \ldots, \underbrace{\mathbf{e}'_N(\mathbf{x}-\mathbf{m})}_{a_N} \rangle$$ ed ho $$\boxed{\mathbf{x} = \mathbf{m} + a_1\mathbf{e}_1 + a_2\mathbf{e}_2 +, \ldots, + a_N\mathbf{e}_N}$$

- Le ricostruzioni, **g**, possono avvenire anche nei sottospazi descritti dalle eigenfaces con eigenvalues più grandi – riduzione dimensionalità

x

$\rightarrow$



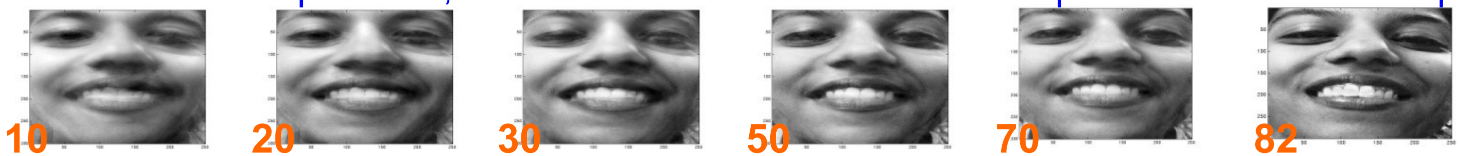
$$\mathbf{m} + a_1\mathbf{e}_1 + a_2\mathbf{e}_2 + a_3\mathbf{e}_3 + a_4\mathbf{e}_4 + a_5\mathbf{e}_5 + a_6\mathbf{e}_6 + a_7\mathbf{e}_7 + a_8\mathbf{e}_8 \ldots + a_N\mathbf{e}_N$$

$\mathbf{g} \approx \mathbf{x}$

x

Se i vettori sono in ordine di importanza, fermandomi all'n-esimo autovettore posso ottenere comunque una buona rappresentazione:

n. coeff.

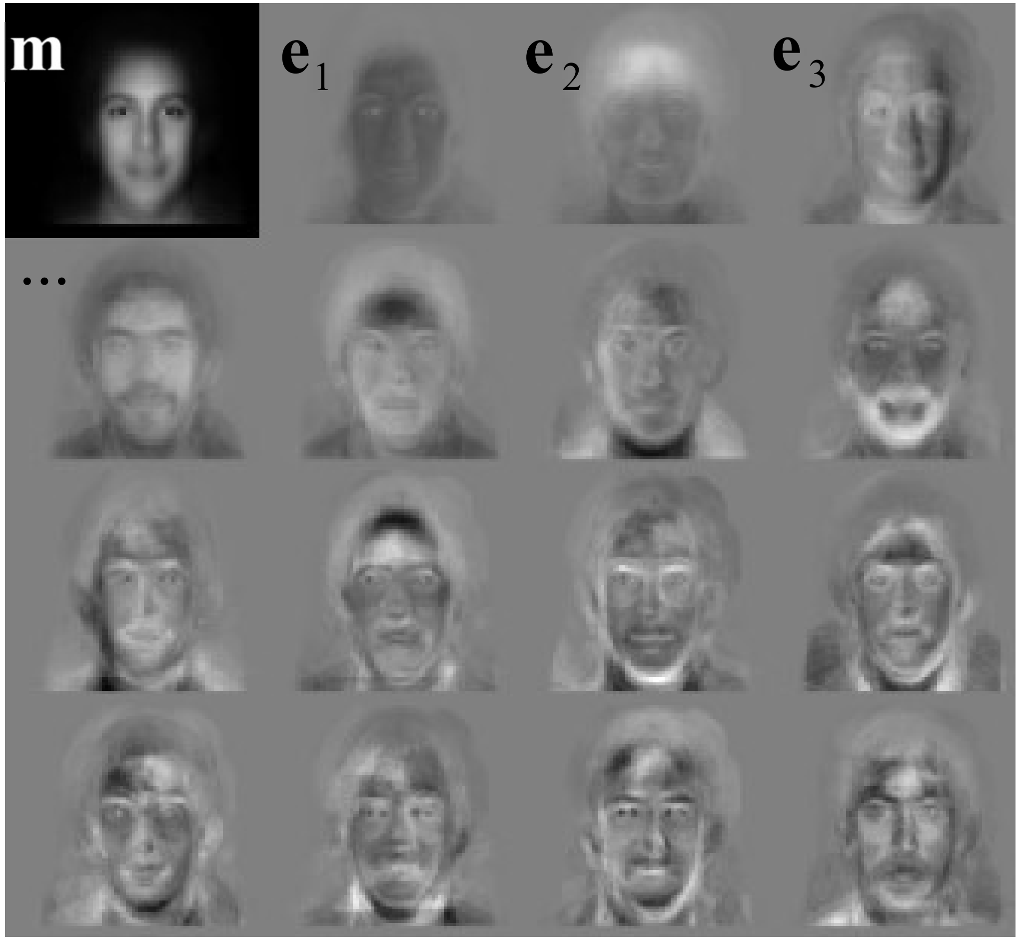


Gli autovettori sono quindi basi che però sono immagini a tutti gli effetti. (in effetti è logico, se una colonna è un viso, un autovettore lo è a sua volta, come non pensarci prima?)

Ogni eigvec rappresenterà in modo più dettagliato una caratteristica, es sopra 5 il naso, 7 le fossette, 6 le palpebre ecc...

Eigenfaces

- Autovettori {**e**} (eigenfaces): in pratica?



es. e7 caratterizza la barba. Se ricostruendo un'immagine ho che a7 è molto grande vorrà dire che la persona di quell'immagine ha la barba (EUREKA!!!)

Eigenfaces - problema

- Sia $\underset{N \times M}{\mathbf{A}} = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(M)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(M)} \\ \vdots & \vdots & \vdots & \vdots \\ x_N^{(1)} & x_N^{(2)} & \dots & x_N^{(M)} \end{bmatrix}$ pertanto **S = AA'** con S come in slide 6!

  N x N

  Dove A è una matrice che concatena di vettori raster scan Nx1 delle immagini di partenza a cui ho sottratto la media m per ogni immagine.

- Con un'img 256 x 256 ho $\underset{65536 \times 65536}{\mathbf{S}}$: problemi di overflow!

- Trick: calcolo gli autovettori di $\boxed{\underset{M \times M}{\mathbf{A}'\mathbf{A}}}$, ossia $\{\tilde{\mathbf{e}}\}$, tenuto conto che di solito si usano 20-30 eigenfaces e che

Dato A NxM, la costruzione di S NxN mi porta in overflow. Quindi creo S' dato da A'A che è MxM e ricavo i suoi autovettori.

Se=λe è impraticabile per troppi pixel. Sostituisco S:

$$\mathbf{A}'\mathbf{A}\tilde{\mathbf{e}} = \lambda\tilde{\mathbf{e}} \xrightarrow{xA} \mathbf{A}\mathbf{A}'\mathbf{A}\tilde{\mathbf{e}} = \lambda\mathbf{A}\tilde{\mathbf{e}}$$

$$\rightarrow \mathbf{S}\mathbf{A}\tilde{\mathbf{e}} = \lambda\mathbf{A}\tilde{\mathbf{e}}$$ ===> Se = λe per Ae_tilde=e quindi

$$\boxed{\mathbf{e} = \mathbf{A}\tilde{\mathbf{e}}}$$

La tilde è data dal fatto che voglio differenziare perché ovviamente otterrò degli autovalori/autovettori diversi. Ma che hanno rapporto con quelli di partenza.

Il rapporto tra e ed e_tilde è una proiezione lineare che è uguale ad A e mi porta da e_tilde ad e.

Eigenfaces - note

- Gli M autovalori di **A'A** corrispondono agli M autovalori più grandi di **S** (così come i corrispondenti autovettori)

Eigenfaces - algoritmo

1. Sottraggo ad ogni immagine $\mathbf{x}^{(k)}$,k=1...M, la media $\mathbf{m} = \frac{1}{M}\sum_{k=1}^{M}\mathbf{x}^{(k)}$ ottenendo i vettori $\left\{\tilde{\mathbf{x}}^{(k)}\right\}$ da cui costruisco la matrice $\underset{N \times M}{\mathbf{A}}$
2. Calcolo M autovettori $\left\{\tilde{\mathbf{e}}_i\right\}_{i=1,\ldots,M}$ di $\underset{M \times M}{\mathbf{A}'\mathbf{A}}$ formando la matrice $\underset{M \times M}{\mathbf{V}}$
3. Ricavo gli M autovettori più grandi di **S** = **AA'**, ossia $\mathbf{e}_i = \mathbf{A}\tilde{\mathbf{e}}_i$

   o in forma matriciale $\underset{N\times M}{\mathbf{U}} = \underset{N\times M}{\mathbf{A}}\ \underset{M\times M}{\mathbf{V}}$
4. Ricavo i corrispondenti componenti (o coeff. di proiezione)

$$a_i^{(k)} = \mathbf{e}'_i\left(\tilde{\mathbf{x}}^{(k)}\right)$$

o in forma matriciale

$$\underset{M\times 1}{\boldsymbol{\omega}}^{(k)} = \underset{M\times N}{\mathbf{U}}'\left(\underset{N\times 1}{\tilde{\mathbf{x}}}^{(k)}\right)$$

Proprietà chiave della rappresentazione ad eigenfaces

Date

- 2 immagini $\mathbf{x}_1, \mathbf{x}_2$ usate per costruire l'eigenspace
- $\boldsymbol{\omega}_1$ è la proiezione nell'eigenspace dell'img $\mathbf{x}_1$
- $\boldsymbol{\omega}_2$ è la proiezione nell'eigenspace dell'img $\mathbf{x}_2$

allora,

$$\boxed{\|\boldsymbol{\omega}_2 - \boldsymbol{\omega}_1\| \approx \|\mathbf{x}_2 - \mathbf{x}_1\|}$$

ossia, la distanza nell'eigenspace è approssimativamente uguale alla distanza tra due immagini.

Riconoscimento con le eigenfaces

TRAINING

1. Analizza il database d'immagini etichettato (<volti,identità>)

   a) Esegui PCA — calcola le eigenfaces, formo **U**
   b) Calcola i K coefficienti per ogni immagine $\mathbf{x}^{(k)}$ k=1,...,M
   c) Calcola le M proiezioni nello spazio delle eigenfaces $\omega^{(k)}$ k=1,...,M
   d) Calcolo la soglia

$$\theta = \max\left\{ \left\| \boldsymbol{\omega}^{(j)} - \boldsymbol{\omega}^{(k)} \right\| \right\} \text{per } j,k = 1,\dots,M$$

RICONOSCIMENTO

2. Data una nuova img (da riconoscere) **x**,

   a) Ne sottraggo la media del dataset di training **m**, ottengo $\tilde{\mathbf{x}}$
   b) Proietto, ossia calcolo le K componenti

$$\boxed{\tilde{\mathbf{x}} \rightarrow \boldsymbol{\omega} = \left[a_1, a_2, \dots, a_K\right]' \quad \text{ossia} \quad \boldsymbol{\omega} = \mathbf{U}'\tilde{\mathbf{x}}}$$

   c) Calcolo il set di distanze

$$\left(\varepsilon^{(k)}\right)^2 = \left\| \boldsymbol{\omega} - \boldsymbol{\omega}^{(k)} \right\|^2 \text{ per } k = 1,\dots,M$$

Il valore di distanza mi dice che ho un test e lo confronto con le altre identità (es. noi avevamo 41 identità, facciamo un paragone tra tutte)

RICONOSCIMENTO

Riconoscimento con le eigenfaces

3. Ricostruisco la faccia usando eigenfaces e componenti

$$\tilde{\mathbf{g}} = \sum_{i=1}^{K} a_i \mathbf{e}_i \quad \text{oppure} \quad \tilde{\mathbf{g}} = \mathbf{U}\boldsymbol{\omega}$$

4. Calcolo la distanza tra la faccia di partenza incognita e la ricostruzione

$$\xi^2 = \left\| \tilde{\mathbf{g}} - \tilde{\mathbf{x}} \right\|^2$$

5. Se
   - $\xi \geq \theta$ non è una faccia
   - $\xi < \theta$ e $\varepsilon^{(\mathrm{k})} \geq \theta, (\mathrm{k} = 1,\ldots,\mathrm{M})$ è una nuova faccia
   - $\xi < \theta$ e $\min \left\{ \varepsilon^{(k)} \right\} < \theta$ è una faccia conosciuta, la k_{best}-esima, dove

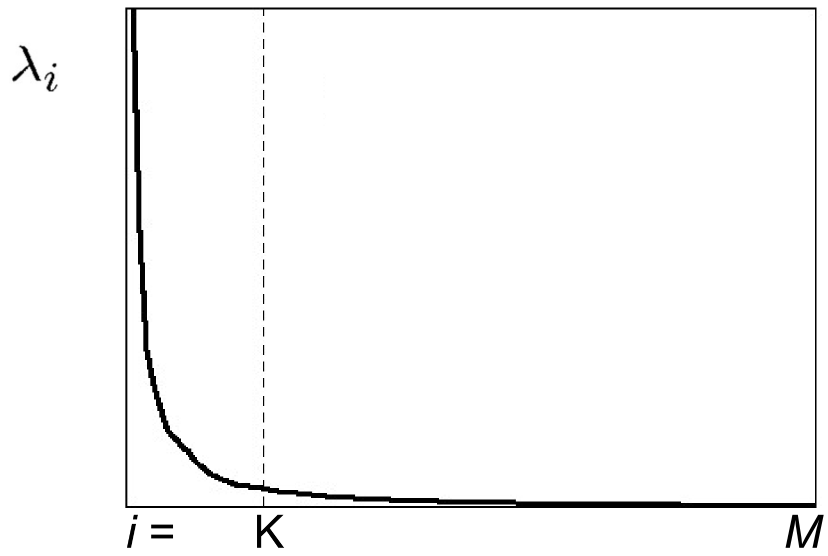
La distanza esistente nel mio spazio è maggiore della massima distanza del mio spazio ridotto

La ricostruzione (csi) è minore, quindi vuol dire che effettivamente ho un volto

$$k_{\text{best}} = \underset{k}{\operatorname{argmin}} \left\{ \varepsilon^{(k)} \right\}$$

Dettagli pratici

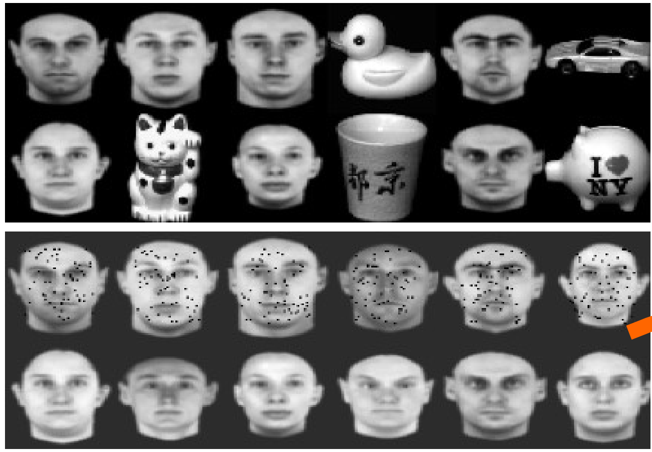
- Quanti eigenvector usare?
- Controlla il decadimento degli eigenvalues $\{\lambda_i\}$
- Dati "buoni" ossia trattabili hanno poche dimensioni ad alta varianza
- Nel caso in cui tutti gli N autovalori sono stati calcolati per un dataset N-dimensionale vale

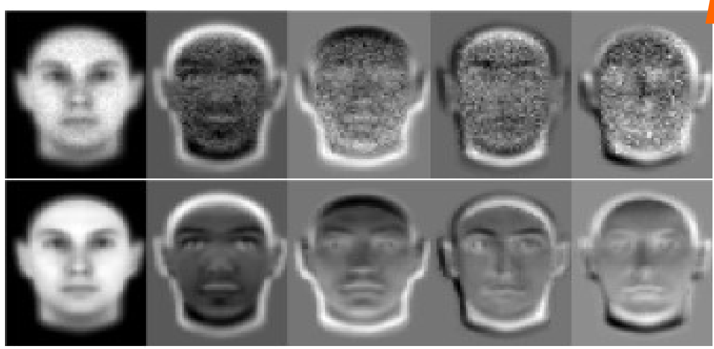


$$\sigma_{\text{covered}} = \frac{\sum_{i=1}^{K} \lambda_i}{\sum_{j=1}^{N} \lambda_j}$$

Problemi eigenfaces

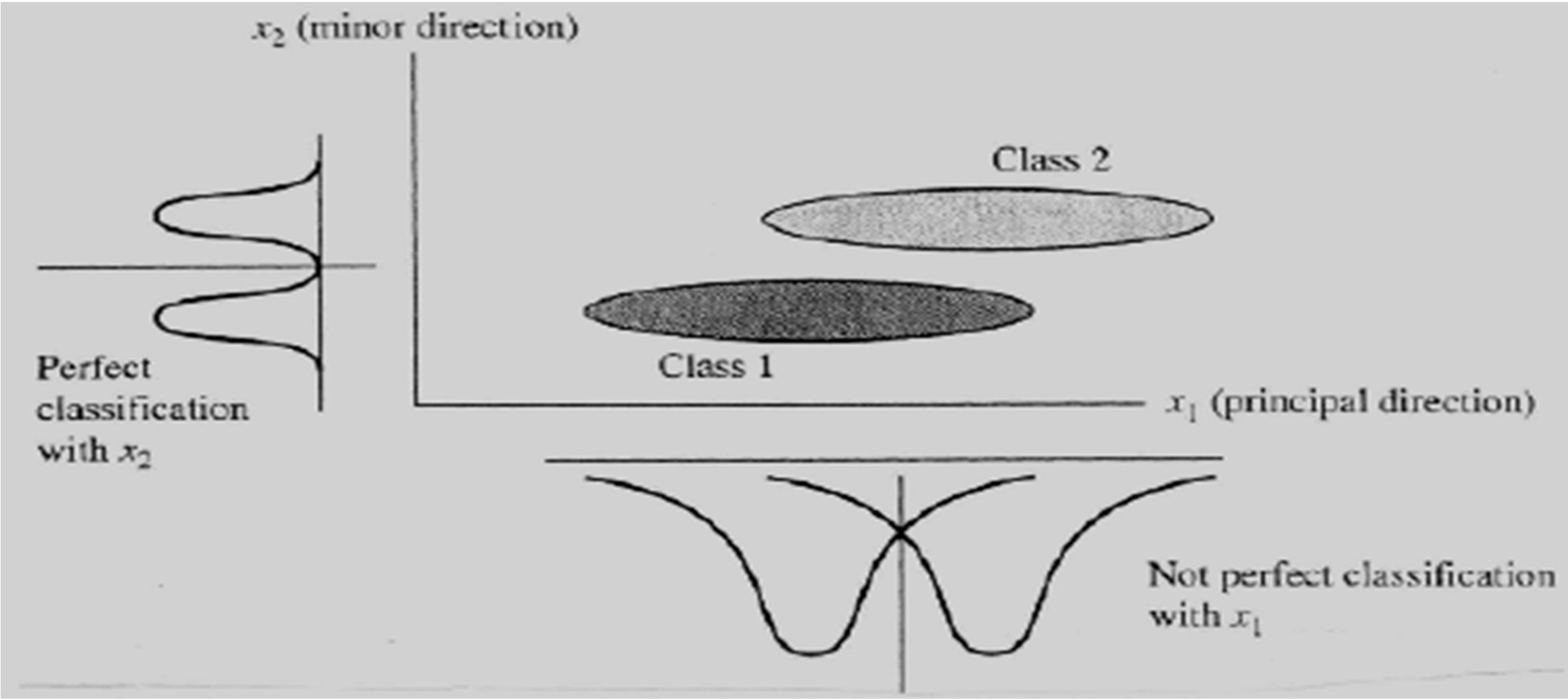
- **Illuminazione**: stessi soggetti con differente illuminazione risultano lontani nello spazio delle eigenfaces
- **Pose differenti** :le componenti di un volto frontale non servono per un riconoscimento con profili
- **Allineamento differente**
- **Espressione facciale differente**
- **Outlier**
  - Sample outliers = non facce
  - Intra-sample outliers = facce affette da rumore





Problemi eigenfaces (2)

- Funzionale solo per la rappresentazione di dati appartenenti ad un'unica classe
- Non separano classi differenti



Riferimenti su libro

- PCA capitolo Duda 3.8.1
- Materiale di approfondimento (paper eigenfaces VS fisherfaces)
- Per linear discriminant analysis, 3.8.2, 3.8.3